NASA / TM—1999-209058 ICOMP–99–04

NPARC v3.1 User's Guide: A Companion to the NPARC v3.0 User's Guide

Joongkee Chung
Institute for Computational Mechanics in Propulsion, Cleveland, Ohio

John W. Slater
Glenn Research Center, Cleveland, Ohio

Ambady Suresh and Scott Townsend
NYMA, Inc., Brook Park, Ohio

March 1999

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized data bases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA / TM—1999-209058 ICOMP–99–04

NPARC v3.1 User's Guide: A Companion to the NPARC v3.0 User's Guide

Joongkee Chung
Institute for Computational Mechanics in Propulsion, Cleveland, Ohio

John W. Slater
Glenn Research Center, Cleveland, Ohio

Ambady Suresh and Scott Townsend
NYMA, Inc., Brook Park, Ohio

National Aeronautics and
Space Administration

Glenn Research Center

March 1999

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

NPARC v3.1 User's Guide: A Companion to the NPARC v3.0 User's Guide

Joongkee Chung*, John W. Slater†, Ambady Suresh ‡, and Scott Townsend§
NASA Glenn Research Center
Cleveland, Ohio 44135

Abstract

NPARC v3.1 is a modification to the NPARC v3.0 computer program which expands the capabilities for time-accurate computations through the use of a Newton iterative implicit method, time-varying boundary conditions, and planar dynamic grids. This document discusses some of the changes from the NPARC v3.0, specifically: changes to the directory structure and execution, changes to the input format, background on new methods, new boundary conditions, dynamic grids, new options for output, usage concepts, and some test cases to serve as tutorials. This document is intended to be used in conjunction with the NPARC v3.0 user's guide.

Contents

*Senior Research Associate, Institute for Computational Methods in Propulsion (ICOMP), Ohio Aerospace Institute.
†Aerospace Engineer,NASA Glenn Research Center, Inlet Branch.
‡NYMA, Inc.
§Computer Engineer, NYMA, Inc.

1 Introduction

This document discusses modifications to NPARC v3.0 which occurred to create NPARC v3.1 and is intended to be a companion to the user's guide for NPARC v3.0 [8]. The procedures discussed in the NPARC v3.0 user's guide are assumed to be applicable unless discussed differently here.

NPARC v3.1 was developed in order to implement several modifications of the NPARC program that were never implemented into NPARC v3.0 and were of specific interest to multi-disciplinary efforts at the NASA Glenn Research Center. These modifications focus on the capability to solve unsteady flows and include the implementation of: 1) a Newton iterative method for implicit, time-accurate computations, 2) capabilities allowing for dynamic planar grids, 3) the Chung-Cole compressor face boundary condition, 4) time-varying freestream and compressor face boundary conditions, and 5) capabilities for the output of unsteady data. As can be seen, the added capabilities derive from a focus on solving unsteady flow in high-speed inlets.

The dynamic grid capability assumes that in the regions of the moving grid, the grid is planar, "quasi-2d", or axisymmetric. A "quasi-2d" grid is one in which the constant-"l" grid surfaces are planar and at constant z-coordinates. To define the moving grid problem, one specifies segments of boundary grid that may move as a rigid body or deform. The grid near the moving segment deforms to accommodate the motion. Thus, the grid requires regeneration at each time step as the segment moves. The flow equations sense the grid motion through the grid speeds, which are computed from a forward time difference of two consecutive grids. The grid motions are considered moderate in that the topology of the block is not destroyed.

The following sections discuss the modifications in NPARC v3.1 and changes and extensions of its usage.

2 Installing NPARC Version 3.1

NPARC v3.1 is installed by obtaining the compressed tar file, untarring it, and then building via `make`:

```
zcat nparc3r1.tar.Z | tar xvf -
cd nparc3r1
make
```

During the build process the script `getconfig` will create an installation specific configuration file `config.mk`. This file specifies the machine architecture, communication library, and default executables to be built. The machine architecture is used to select `config/$ARCH.mk`, while the communication library is used to select `config/$COMMLIB.mk`. If for some reason the standard build procedure fails, editing these files to better reflect the local host's configuration may fix the problem.

Note: the default build procedure does not attempt to build various X11/Motif based utilities which are located in the `util` subdirectory. If you wish to build these, `cd` to the directory and type `make`.

This release has been built on Cray, HP, IBM, Linux, SGI, and Sun systems using Oak Ridge PVM, MPICH/P4, SGI MPI, and SGI PVM communication libraries. DEC Alpha and HP/Convex CSPP configuration files are available, but have not been tested.

The command `make help` will list the available `make` targets. Of these, the `reconfig` target is unusual in that it expects `make` macro specifications such as:

```
make reconfig OPTFLAGS="-g -64 -mips4"
```

This particular example would cause a rebuild with the specified compiler options. Reconfiguring the communication library is done similarly:

```
make reconfig COMMLIB=mpich
```

Equivalently, one could just edit `config.mk`, and then execute `make clobber` and then `make`.

3 Directory Structure for NPARC v3.1

The directory structure of NPARC v3.0 has been simplified for NPARC v3.1. The primary goal was to retain the single directory feel of NPARC v2.0 while still supporting the shared 2D and 3D source capability of NPARC v3.0. With the new structure all 2D (3D) source code operations (searching, editing, compiling, etc.) can be done within the `2d` (`3d`) subdirectories. The directories and their contents are listed below:

. The top-level makefile is here, along with machine independent `make` 'include' files and the `getconfig` shell script. Invoking '`make kit`' here will create a compressed `tar` file containing the full distribution of the code.

2d, 3d 2D (3D) specific code and symbolic links to the common code. This is where the NPARC executable files are built. Invoking '`make kit`' here will create a compressed `tar` file containing just the files necessary to build a 2D (3D) version of the code.

bin Symbolic links to the various executable files.

common Code common to both the 2D and 3D versions of NPARC. These files will have symbolic links in the `2d` and `3d` directories pointing at them.

config Machine and communication library dependent `make` 'include' files and `m4` files. The `m4` files are used to manage the FORTRAN $\leftrightarrow$ C linkage mechanism on the various machines. This directory will have symbolic links pointing to it from most other directories. These links make the full distribution and the 2D/3D only distributions look the same to the makefile scripts.

doc This document, in both PostScript and LaTeX source form. Updated NPARC quick guides, in both PostScript and `groff 1.09` source form. NPARC v3.0 user and installation documents in PostScript form. The manual page for `runnparc` in formatted and `nroff` source form. Informal descriptions of version control operations, build procedures, and the VCE implementation.

tests This directory is the root of a set of test cases. Each unique case is in its own subdirectory. Invoking '`make test`' will run the tests on the local host, serial and/or parallel depending upon which NPARC executables have been built.

tools Various version control scripts layered on top of SCCS (Source Code Control System). Also LSF `echkpnt` and `erestart` scripts to support checkpointing and restarting serial or parallel runs. None of these tools are necessary to use NPARC.

util NPARC v3.0 utilities `icees`, `memsize`, `prenparc`, and `reynold`.

4 Running NPARC v3.1

Running NPARC v3.1 is done in much the same manner as NPARC v3.0 with the exceptions as noted within this document.

NPARC v3.1 incorporates new checkpointing and shutdown capabilities into both the serial and parallel versions of the code, as well as new facilities added to `runnparc` to aid in running the parallel code in special situations. Aside from the new features listed here, NPARC v3.1 is run in the same manner as NPARC v3.0.

An NPARC *checkpoint* performs the same operations that are done every `NP` iterations. The intent is to save the current computation state so that it may be restarted in the future, possibly in response to some host failure. A checkpoint can be initiated by either sending `SIGUSR1` to the master process or creating a file named `NPARC_CKPT` in the working directory of the master process. Note that due to synchronization issues, the parallel version will take a couple iterations before actually beginning to write the checkpoint files. At the end of the checkpoint operation the `NPARC_CKPT` file will be removed.

NPARC may now be shut down in a controlled manner before `NMAX` iterations have been completed. The shutdown may be initiated by either sending `SIGUSR2` to the master process or creating a file named `NPARC_STOP` in the working directory of the master process. Note that due to synchronization issues, the parallel version will take a couple iterations before actually beginning to write the output, restart, and optional Plot3D files.

The `runnparc` script has been updated in a number of ways to aid in running the code in special situations:

- You can now specify a directory for the 2D (3D) executables via the environment variable `NPARC_2D` (`NPARC_3D`). This is helpful when the machines in a parallel run are different architectures but use the same file server.
- Through the `-force` option it is now possible to force `runnparc` to use the given hosts exactly as specified. Normally `runnparc` uses the list of hosts as candidates which are then scanned to determine the number of processes to assign to a host. The 'force hosts' option disables `runnparc`'s internal host assignment mechanism. Note that it is an error to use `-force` when the first host listed is not the local host.
- Some systems do not handle NPARC's convention of input from `stdin` and output to `stdout` well. These systems are now supported through the `-input` *file* and `-output` *file* options.
- To handle situations too complex for `runnparc`'s automatic setup methods, it is now possible to explicitly specify the user name, host name (or alias), working directory, and path to the executable for each process. The information is specified via a file and passed to `runnparc` via the `-procdef` *file* option.
- The `-Restart` option is used to restart a checkpointed computation.

- `runnparc`'s support for MPI libraries has been improved, particularly for the MPICH/P4 implementation which now has the same level of support as Oak Ridge PVM. If a `-hosts` or `-procdef` option is provided, `runnparc` will create a P4 *procgroup* file either through the normal automatic setup mechanism or from the information supplied in the process definition file. Otherwise, `runnparc` simply invokes `mpirun`, assuming it will perform all necessary process assignment.
- Primarily due to lack of access, `runnparc` does not support the DQS or NQS batch queueing systems. Support has also been dropped for the EZP, MPL, PVMe, and T3D communication libraries.

5 Getting Help for NPARC Version 3.1

In general, for help with NPARC one should contact the User Support as listed in the NPARC v3.0 User's Guide. However, for specific questions regarding features added in NPARC v3.1, one can directly contact the developers at NASA Glenn. For questions regarding the new directory structure and changes with running NPARC in parallel mode, one can contact Scott Townsend at (216) 977-1080 or s.townsend@grc.nasa.gov. For questions regarding the time-accurate and moving grid capabilities, one can contact John Slater at John.Slater@ grc.nasa.gov or (216) 433-8513. For questions regarding the compressor face boundary condition, time-accurate, and unsteady output capabilities, one can contact J.K. Chung at jkchung@grc.nasa.gov or (216) 433-2411. The questions regarding using NPARC with the Visual Computing Environment (VCE), one can contact Ambady Suresh at (216) 977-1384 or fsbern8@jupiter.grc.nasa.gov.

6 Time Accuracy

The most significant modification of NPARC v3.0 has been the added capability to solve unsteady flow problems in a second-order, time-accurate manner. The following sections discuss some of these modifications.

6.1 Solver Options (`ISOLVE`)

For NPARC v3.1, the options for the solver flag `ISOLVE` includes:

```
ISOLVE   =0  Generate time-varying grid only
         =1  Pentadiagonal solver (default)
         =2  Newton iterative implicit method
         =3  3-stage pseudo-Runge-Kutta solver
         =4  4-stage pseudo-Runge-Kutta solver
         =5  5-stage pseudo-Runge-Kutta solver
```

The option `ISOLVE = 0` was added to allow for the generation of the time-varying grid (`IGRDYN = 2`) without the computation of the flow solution. This allows one to verify that the grid motion and regeneration is being performed correctly without the added computational expense of the flow computation. One can use this option with the output of the time-varying grid files (positive `NQFREQ`) to see snapshots of the grid a various times. Since the flow is not being computed, one can choose any convenient time step. It is advised to use `IVARDT = 0` with a convenient time step specified by `DTCAP`. Also, values of 1.0 are displayed in the screen output for `L2 RESIDUAL`.

A new option exists (`ISOLVE`=2) to use the Newton iterative method to perform implicit sub-iterations at each time step to reduce errors and yield a nominally second-order time accuracy.

The Runge-Kutta solvers (`ISOLVE = 3,4,5`) can also be used to obtain second-order time accuracy; however, the CFL condition may limit the allowable time step. The Newton iterative method allows a larger time step, although it is more computationally expensive. Thus by taking larger time steps, one hopes to reduce the number of iterations required to compute the flow, and so, reduced to overall computational expense.

6.2 Newton Iterative Method

A Newton iterative method allows for a nominally second-order, time-accurate implicit time integration. For the Newton iterative method, the iterative equation takes the form

$$\left[I + \frac{1}{c_0} \frac{\partial \hat{A}^m}{\partial \xi_j} \right] \Delta \hat{Q}^m = - \left(\hat{Q}^m + \frac{c_1}{c_0} \hat{Q}^n + \frac{c_2}{c_0} \hat{Q}^{n-1} \right) - \frac{1}{c_0} \hat{F}^m. \tag{1}$$

where c_0, c_1, and c_2 depend on the grid spacing and time step. If the time step is uniform, $\Delta\tau_1 = \Delta\tau_2 = \Delta\tau$, where

$$\Delta\tau = t^{n+1} - t^n \tag{2}$$

then

$$c_0 = \frac{3}{2\Delta\tau}, \quad c_1 = -\frac{4}{2\Delta\tau}, \quad \text{and} \quad c_2 = \frac{1}{2\Delta\tau}.$$

This equation can be iterated until the right-hand-side becomes zero, which is the discretized form of the Navier-Stokes equations,

$$\hat{Q}_\tau + \hat{R} = 0.$$

Further details can be found in Ref. [13].

The Newton iterative method is used in NPARC when `ISOLVE = 2`. The maximum number of subiterations that may be performed at each time step is controlled by the input variable `NSUBMX`. Typically using a value of `NSUBMX = 3` provides for good convergence of the subiterations. The default value is `NSUBMX = 1`. The subiterations performed at each time step will end if the subiteration residual drops below a value specified by `TOLSUB` input variable.

The value displayed as `L2 RESIDUAL` in the convergence output is the residual of the right-hand side after the subiterations. For a computation converging to steady-state, this value will drop as expected; however, for an unsteady computation, this number should be "low" and remain fairly uniform throughout an unsteady computation. A 'high' value indicates that perhaps more sub-iterations per time step are required (i.e. increase NSUBMX). Conversely a value near machine zero may indicate that perhaps NSUBMX could be reduced to save computational time. One can also reduce TOLSUB to limit the number of sub-iterations.

The subiterations are applied for each grid block. Ideally, each subiteration should be done for all the blocks to assure accurate communication between blocks. There may be a lag in the update of the flow at the boundaries of the blocks and this will adversely effect the time accuracy if the boundary is in a region of quickly changing flow conditions. The subiterations could be applied over each sequence through the blocks. The applications we have studied (which have involved much of the unsteady flow occurring in a single block) have not required this multi-block subiterations. Thus the Newton subiterative method should be used with caution for multi-block problems.

6.3 Real-Time

For NPARC v3.0, the inputs, computations, and output are all in nondimensional units. Since NPARC v3.1 is concerned with unsteady flow and time-varying conditions, the capability has been added to read some time input variables and write some time outputs in time units of seconds. The input variable `IREALT` specified in the `INPUTS` namelist is a flag specifying how time parameters are read and written:

```
IREALT   Flag controlling units for time
         (0) Time parameters are nondimensional (default)
         (1) Time parameters are dimensional in units of seconds
```

When `IREALT = 1`, a reference time is computed as

$$t_{ref} = L_{ref} \; / \; a_{ref}. \tag{3}$$

The L_{ref} is the reference length specified by the input variable `REFLEN` and has a default value of 1.0 ft. The a_{ref} is the reference velocity, which is the acoustic velocity computed as

$$a_{ref} = (\gamma \, R \, T_{ref})^{1/2} \tag{4}$$

The R is the gas constant specified by the input variable GASCON and has a default value of 1716.0 ft lb / (slug R). The T_{ref} is the reference temperature specified by the input variable TREFR and has a default value of 500 degree Rankine. The input variable TREFR is also used to dimensionalize output as described in the NPARC v3.0 user's guide. The input variables REFLEN and GASCON are only used to dimensionalize time, and so, they can be any consistent units.

When IREALT = 1, the input variables TSTART and TFINAL are assumed to be in units of seconds; otherwise for IREALT = 0, they are assumed to be non-dimensional time units.

All solution files output non-dimensional time, except for the shock position output as specified through ISFREQ.

6.4 Time Control

Time becomes important when solving unsteady flows. The input variables TSTART and TFINAL were added to the INPUTS namelist block to indicate the physical times for the start and end of the flow computation. The common variable TOTIME is the accumulated time for the computation. The time-marching proceeds until TOTIME reaches TFINAL or the number of time steps reaches NMAX. The restart file is assumed to contain the grid and solution at TSTART. A new restart format is introduced (see below) in which the current time is written into the file. The time written in the restart file is used as the starting time if TSTART is not specified or is negative. When IREALT = 0, the values of TSTART and TFINAL should be non-dimensional. Likewise, when IREALT = 0, the values of TSTART and TFINAL should be in units of seconds.

6.5 Time Step

The time step for a time-accurate computation must be uniform over the entire flow domain. The size of the time step is not only limited by stability considerations, but also by time-accuracy considerations. One should use the maximum time step possible at each time step for efficient computation. NPARC v3.1 adds more options for the IVARDT input variable:

```
IVARDT   =0  Global DT = DTCAP
         =1  Local DT = DTCAP / ( 1 + sqrt(J) )
         =2  Local DT = DTCAP * VARDT(J,K,L)  (default)
         =3  Same as 2, but DT is increased in boundary layers
         =4  Global DT = DTCAP * [ VARDT(J,K,L) ]_(max dQ)
         =5  Global DT = DTCAP * min[ VARDT(J,K,L) ]_(all blocks)
         =6  Same as 5 but uses convective condition
```

Within the code, the non-dimensional time step used at each grid point is computed with the formula

$$\Delta\tau_{(j,k,l)} \;=\; DT \;*\; VARDT(J,K,L) \tag{5}$$

where the value and meaning of DT depends of the value of IVARDT. VARDT is the array for spatial variation of the time step for the local time-stepping.

For IVARDT = 0, the time step is uniform and constant for all grid points of all blocks. When IREALT = 0, the non-dimensional time step is input through the variable DTCAP and DT = DTCAP, Also, VARDT(J,K,L) = 1.0 for the entire domain. When IREALT = 1 the dimensional time step is input through the variable REALDT and is input in units of seconds.

For IVARDT=1, 2, & 3, local time stepping is used to enhance convergence for steady-state computations. These options should only be used for non-time-accurate computations. For each of these options, DTCAP is the CFL number. For IVARDT = 1, a Jacobian scaling is used to define VARDT to adjust the local time step, which is more efficient than computing the eigenvalues.

$$VARDT(J,K,L) \;=\; \frac{1.0}{1.0 \;+\; J^{1/2}}. \tag{6}$$

For IVARDT = 2, VARDT is set to the inverse of the maximum absolute value of the acoustic eigenvalue

$$VARDT(J,K.L) \;=\; \frac{1.0}{|\;\lambda_{acoustic}\;|_{max}} \tag{7}$$

where

$$\lambda_{acoustic} = |U| + a\ S \tag{8}$$

where U is the contravariant velocity, S is the L_2 norm of the metrics. For the ξ-direction, these are

$$U = \xi_t + \xi_x\ u + \xi_y\ v + \xi_z\ w \tag{9}$$

and

$$s = (\xi_x^2 + \xi_y^2 + \xi_z^2)^{1/2}. \tag{10}$$

Similar expressions can be applied in the η-direction. For viscous flows, a viscous correction is added which further reduces the time step.

For `IVARDT = 3` the `VARDT` is computed the same as for `IVARDT = 2`, but the time step is increased in the boundary layers since it was found from numerical experiments that the time step could be increased without endangering stability.

For `IVARDT = 4`, a global time step is defined by using the `VARDT` at the location of the maximum change in Q. The `DTCAP` is the CFL number.

The option `IVARDT = 5` is new in NPARC v3.1 and uses the minimum `VARDT` of all the blocks to compute `DT` and then sets `VARDT(J,K,L) = 1.0` for all grid points. The `DT` is used as the global value for all grid points to allow for time accuracy.

For unsteady flows solved with an implicit method which may have a significant stability range, the CFL limit may be too restrictive. However, it is felt that there is a convective limit on the time step size for time-accuracy. A time step can be defined by

$$\Delta\tau_{(J,K,L)} = 1\ /\ max(|U|, |V|, |W|) \tag{11}$$

This restriction simply limits the time step such that a fluid particle will not traverse a distance greater than the length of the cell. This approach for determining the proper step size is available as the input option `IVARDT = 6`.

7 Dynamic Grids

NPARC v3.1 is capable of solving for the unsteady flow with dynamic grids. The flow equations and boundary conditions account for the grid motion through inclusion of the velocity components of the grid points, which are the grid speeds. Within NPARC v3.1, the manner in which the grid speeds are determined is indicated by the input variable `IGRDYN`. A value of `IGRDYN = 0`, which is the default, indicates that all grid speeds are zero, which means the grid is static. A value of `IGRDYN = 1` indicates that the grid speeds are specified within the restart file and remain constant throughout the computation. Thus the grid undergoes a rigid-body translation and / or rotation. No deformation of the grid occurs and no grid regeneration is required. A value of `IGRDYN = 2` indicates that one or more segments of the boundary grid may be in motion, and so, a deformation of the grid may occur. The grid may be regenerated over a time step and the grid speeds are computed from a forward time-difference of the grids at the two time levels.

To be able to use the option `IGRDYN = 2`, the grid must be planar, which is the case for two-dimensional or axisymmetric flow problems. For three-dimensional flow problems, the grids must be "quasi-2d" or axisymmetric. A "quasi-2d" grid means that the grid surfaces in the k-direction must be planar and at constant z-coordinates. For axisymmetric grids, the grid surfaces in k-direction must be planar and at constant θ or circumferential angles about the x-axis.

Further details on defining the dynamic grid problem and providing proper input to NPARC v3.1 are discussed below. In the next few sections, the flow equations are discussed with respect to the accounting of the grid motion.

7.1 Conservation Equations for Dynamic Grids

NPARC v3.1 solves the governing conservation equations in an absolute frame of reference. The equations are discretized as a the finite-difference form of the equations in strong conservation form with transformations

between physical and computational space. The strong conservation form of the Navier-Stokes equations is

$$\frac{\partial \hat{Q}}{\partial \tau} + \hat{R} = 0 \tag{12}$$

where

$$\hat{Q} = Q / J \tag{13}$$

and

$$\hat{R} = \frac{\partial \hat{F}_j}{\partial \xi_j}. \tag{14}$$

The generalized flux components are

$$\hat{F}_j = \frac{1}{J}\left(\frac{\partial \xi_j}{\partial t} Q + \frac{\partial \xi_j}{\partial x_k} F_k \right) \tag{15}$$

where

$$\xi_j \in (\xi, \eta, \zeta) \quad \text{and} \quad x_j \in (x, y, z).$$

The $\partial \xi_j / \partial t$ are the time metrics associated with the time variation of the grid. The contravariant velocities are given by

$$U_j = \frac{\partial \xi_j}{\partial t} + u_k \frac{\partial \xi_j}{\partial x_k} \tag{16}$$

7.2 Temporal and Spatial Metrics

The spatial and temporal metrics required above are defined for the transformations between the (t, x, y, z) boundary-conforming physical space and the (τ, ξ, η, ζ) computational space. The transformations for the three-dimensional system take the form

$$t = t(\tau) = \tau, \qquad x = x(\tau, \xi, \eta, \zeta), \qquad y = y(\tau, \xi, \eta, \zeta), \qquad z = z(\tau, \xi, \eta, \zeta),$$

and conversely,

$$\tau = \tau(t) = t, \qquad \xi = \xi(t, x, y, z), \qquad \eta = \eta(t, x, y, z), \qquad \text{and} \qquad \zeta = \zeta(t, x, y, z).$$

The chain rule expansion of the time derivative is

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial \tau} + \xi_t \frac{\partial}{\partial \xi} + \eta_t \frac{\partial}{\partial \eta} + \zeta_t \frac{\partial}{\partial \zeta}, \tag{17}$$

which shows the significance of the time metrics.

The Jacobian is expressed as

$$J^{-1} = x_\xi \left(y_\eta z_\zeta - y_\zeta z_\eta \right) - y_\xi \left(x_\eta z_\zeta - x_\zeta z_\eta \right) + z_\xi \left(x_\eta y_\zeta - x_\zeta y_\eta \right). \tag{18}$$

The grid speed vector is defined as

$$\vec{g} = x_\tau \, \hat{\imath} + y_\tau \, \hat{\jmath} + z_\tau \, \hat{k}. \tag{19}$$

The time metrics can then be written as

$$\xi_t = - \vec{g} \cdot \nabla \xi = - (\, x_\tau \xi_x + y_\tau \xi_y + z_\tau \xi_z \,) \tag{20}$$

$$\eta_t = - \vec{g} \cdot \nabla \eta = - (\, x_\tau \eta_x + y_\tau \eta_y + z_\tau \eta_z \,) \tag{21}$$

and

$$\zeta_t = - \vec{g} \cdot \nabla \zeta = - (\, x_\tau \zeta_x + y_\tau \zeta_y + z_\tau \zeta_z \,) \tag{22}$$

For two-dimensional or planar representations, the transformation takes the form

$$t = t(\tau) = \tau, \qquad x = x(\tau, \xi, \eta), \qquad y = y(\tau, \xi, \eta),$$

and conversely,

$$\tau = \tau(t) = t, \qquad \xi = \xi(t, x, y), \qquad \text{and} \qquad \eta = \eta(t, x, y).$$

The Jacobian of the transformation is

$$J^{-1} \;=\; x_\xi \; y_\eta \;-\; x_\eta \; y_\xi .$$

The metrics of the transformation are,

$$\xi_t = -(x_\tau \xi_x + y_\tau \xi_y), \qquad \xi_x = \; y_\eta \; J, \qquad \xi_y = -\; x_\eta \; J,$$

$$\eta_t = (x_\tau \eta_x + y_\tau \eta_y), \qquad \eta_x = -\; y_\xi \; J, \qquad \eta_y = \; x_\xi \; J.$$

The metrics for the axisymmetric flow assumption follow those presented above with the additional factor of the radius r from the axis of symmetry to the point at which the metric is evaluated. This distance is usually the Cartesian y-coordinate for the planar grid.

The metrics are evaluated in the same manner as NPARC v3.0, which is a central difference of the grid. Expressions for the spatial metrics can be found in Ref. [5].

7.3 Grid Speeds

The grid speeds are computed from a first-order, forward time difference of the grids at the two consecutive time levels

$$\vec{g} \;=\; \left(\; \vec{r}^{\,n+1} \;-\; \vec{r}^{\,n} \; \right) \; / \; \Delta\tau .$$

7.4 Geometric Conservation Law

The geometric conservation law is a statement that the time rate of change of the volume of a control volume is equal to the change in the integration of the motion of the surface of the control volume [16]. The geometric conservation law is obtained from the differential formulation of the Navier-Stokes equations by the assumption that flow is uniform. The geometric conservation law is

$$(J^{-1})_\tau \;=\; -\; \frac{\partial}{\partial \xi_j} \left[\; \frac{1}{J} \left(\frac{\partial \xi_j}{\partial t} \right) \; \right]. \tag{23}$$

The geometric conservation law becomes a constraint to be satisfied during a dynamic grid computation.

At each subiteration of the Newton iterative method, the value of $\hat{Q}$ is updated. The geometric conservation law (GCL) must be satisfied to properly account for grid motion. Here the approach is to determine a value for V that will decode $\hat{Q}$ to provide the value of Q,

$$Q \;=\; \hat{Q} \; J.$$

For the Newton iterative method, the new Jacobian can be evaluated as

$$(J^{-1})^{n+1}_{GCL} \;=\; \frac{4}{3}(J^{-1})^n \;-\; \frac{1}{3}(J^{-1})^{n-1} \;+\; \frac{2\Delta\tau}{3} \; (J^{-1})^{n+1}_\tau \tag{24}$$

The above expression for $(J^{-1})_\tau$ can be applied.

At each subiteration, the new value of $\hat{Q}$ can be evaluated as

$$\hat{Q}^{n+1} \;=\; \hat{Q}^{n+1} \; / \; ((J^{-1})^{n+1}_{GCL} \; J^{n+1}_{GRID})$$

where J^{n+1}_{GRID} is the Jacobian as computed from the known grid at $n+1$.

When `ISOLVE = 1`, the geometric conservation law (GCL) is applied as

$$(J^{-1})^{n+1}_{GCL} \;=\; (J^{-1})^n \;+\; \Delta\tau \; (J^{-1})^{n+1}_\tau . \tag{25}$$

For the Runge-Kutta method, the geometric conservation law (GCL) is applied as

$$(J^{-1})^{n+1}_{GCL} \;=\; (J^{-1})^n \;+\; \Delta\tau \; (J^{-1})^n_\tau . \tag{26}$$

7.5 Turbulence Modeling and Dynamic Grids

When an algebraic turbulence model is used, dynamic grids will not effect the computation of the turbulent viscosity, μ_t. If a transport equation for turbulence properties is solved as part of the turbulence model, such as for the k-ϵ model, the effects of the dynamic grid on convective flux components may be significant. However, in NPARC, the effects of grid dynamics is not taken into account in the turbulence equations.

7.6 Grid Motion (IGRDYN)

The input variable `IGRDYN` indicates the level of grid dynamics. The options include:

```
IGRDYN  Grid dynamics flag.
        (=0) Grid is static and grid speeds are zero (default).
        (=1) Grid is dynamic and undergoes rigid-body motion.
        (=2) Grid is dynamic and undergoes relative motion.
```

A value of `IGRDYN = 0`, which is the default, indicates that the grid should remain static through the computation. The grid speeds are set to zero and no grid regeneration is performed.

A value of `IGRDYN = 1` indicates that the entire grid can undergo rigid-body translation and / or rotation depending on the values of the grid speeds as read in from the restart file (see below for a detailed discussion on the restart file format when `IRSTFL = 1`). The grid speeds are the absolute velocity components at each grid point as determined from rigid-body dynamics. The values of the grid speeds remain constant throughout the computation and no grid regeneration is performed.

A value of `IGRDYN = 2` indicates that one or more segments of the boundary grid translates, rotates, or deforms relative to the rest of the grid. The parameters defining the dynamics of these segments are specified through an auxiliary input file (`fort.54`). The grid becomes an explicit function of time and grid regeneration is performed at each time step. The grid speeds are computed from a time difference of the grids. Two types of relative motion are possible. The first type is when a segment translates and rotates as a rigid-body relative to the rest of the boundary grid. The second type is when the segment deforms relative to itself and the rest of the boundary grid. The grid regeneration automatically detects which regions of the boundary grid is in motion and only performs the regeneration for those regions, which reduces the computational cost.

7.7 Grid Dynamics Parameter Input File (fort.54)

When `IGRDYN = 2`, NPARC reads file `fort.54` for the input parameters defining the dynamics of the moving boundary grid segments. The grid as read in through the restart file is taken as the state of the grid at the time specified in the restart file. The initial grid speeds are also read in from the restart file. The dynamics of the grid is defined by specifying which segments of the boundary grid are in motion, the type of motion, and schedules for the motion. The segments are specified by listing for each segment, the block and (j,k) range. The segment must be along a constant j or k line and on the boundary. The segment may be one of three types:

1) **Rigid**. The segment translates and rotates about a defined rotation point. The additional inputs required are a schedule defining the translation and rotation with respect to time and a rotation point.

2) **Variable.** This segment can change shape to adjust the domain boundary to the motions of the rigid-body segments. The variable segment is defined by a cubic-spline using end position and tangent conditions. The additional inputs required include the type of end tangency conditions. The options include no end tangency specified, keep the ends tangent to the original segment, keep the ends tangent to the adjacent segments, or keep the ends normal to the adjacent segments (which may be useful at intersections of block faces).

3) **Deforming.** This segment can deform according to a coded relation in the subroutine **deform.f.** Within NPARC v3.1 the subroutine **deform.f** is coded to solve the test case of a flexible bump collapsing in a

annular duct as described in a section below. One can code a different deformation into **deform.f** and then rebuild the NPARC executable.

An example of the grid dynamics input file is the file for a flat plate with a rotating flap:

```
Flat plate with a rotating flap hinged at (0.6,0.0)
Number of dynamic segments
  3
Seg  Type  Block  ibeg  iend  jbeg  jend  Sch/EndBeg  Rotpt/EndEnd
  1     2      1    15    26     1     1           1             2
  2     1      1    26    51     1     1           1             0
  3     2      1    51    51     1    52           3             3

Number of rotation points
  1
Seg       xrot/i       yrot/j
  2          0.6          0.0

Number of Schedules
  1
Number of data points per schedule
  3
Schedule 1:    Time        x        y   Rotation(degrees)
                0.0      0.0      0.0                 0.0
                1.0      0.0      0.0                15.0
                5.0      0.0      0.0                15.0
```

The structure of the lines of the input file in the order in which they must be listed is as follows:

Title. A short description of the grid dynamics problem.

Header. A header for the number of dynamic segments.

Number of segments. An integer defining the number of dynamic segments. In the above example, there are three segments.

Header. A header for the list of dynamic segments.

List of dynamic segments. For each dynamic segment, there are nine integers that are required. The first is the segment identifier. The second integer indicates the type of dynamic segment:

```
Type of dynamic segments
(1) rigid
(2) variable
(3) deforming
```

The third integer is the block in which the segment is located. The fourth and fifth integers define the `J` range of the segment. The sixth and seventh integers define the `K` range of the segment. The eighth and ninth integers are flags whose meanings depends on the type of segment. For a rigid segment, the eighth integer is a flag indicating which schedule is to be used to define the dynamics and the ninth integer is a flag indicating how the rotation point is defined.

```
Type of rotation point
(0) fixed, (x,y) are specified
(1) fixed, (j,k) are specified
(2) translating, original (x,y) are specified
(3) translating, original (j,k) are specified
```

For a variable segment, the flags indicate the end tangency condition at the start and end of the segment, respectively.

```
End tangency condition
(0) no tangency
(1) maintain the original tangent vector
(2) maintain the tangent to the adjacent segment
(3) maintain the normal to adjacent segment
```

For a deforming segment, the two flags are optional inputs.

Blank line. After the list of segments, a blank line is required.

Header. A header for the number of rotation points.

Number of rotation points. An integer defining the number of rotation points. *If there are no rigid segments (type 1 segments), then this number is zero and the next header and list of rotation points should not be included.*

Header. A header for the rotation points. *This line should not be included if there are no rotation points.*

List of rotation points. The rotation points for each rigid segment (type 1 segments) are defined. All rotation points are assumed to be fixed at $(0.0, 0.0)$ unless specified here to be something else. If a rigid segment only translates, the default rotation point can be used and nothing further is needed. The rotation points are listed by segment. The rotation points can be specified by entering the (x, y) coordinates or by entering the (j, k) indices of the rotation point, if it is coincident with a grid point. Further, one must indicate if the rotation point is fixed in space, or allowed to translate as the segment translates. The last entry in the segment input line is a flag to indicate the type of rotation point specified: 0) fixed coordinates with (x, y) specified, 1) fixed coordinates with (j, k) specified, 2) translating coordinates with (x, y) specified, 3) translating coordinates with (j, k) specified. *This list should not be included if there are no rotation points.*

Blank line. After the list of rotation points, a blank line is required.

Header. A header for the number of schedules.

Number of Schedules. An integer defining the number of schedules. *If there are no rigid segments (type 1 segments), then this number is zero and the next header and list of schedules is not needed.*

Header. A header for the number of data points for each of the schedules.

Number of data points per schedule. Each number is listed on the same line separated by spaces (1 or more).

Schedules Each rigid-body segment (type 1 segment) must be assigned a schedule to define the translation and rotations as a function of time. The next series of lines list each schedule in groups according to the number of data points per schedule. The first line of a group is a header line. The lines following (one line for each data point), lists the time, the (x, y) translations, and the rotation (in degrees). The translations and rotations are with respect to some convenient reference. The grid regeneration simply takes a known grid at a known time and regenerates the grid for a new time. Knowing the two time levels and the schedule, the Δx, Δy, and $\Delta\theta$ can be determined and then used to move the grid to the new time level. One should make sure that the start time for the schedule is the same or less than the start time of the computation as specified in `TSTART`. The final time of the schedule should be GREATER than the final time of the computation as specified by `TFINAL` to allow for proper computation of the dynamic grid.

7.8 Example: Flap on a Flat Plate

The file listed above is for a flap on a flat plate. The flap is segment 2 and is a rigid segment. It is defined by the boundary grid point $J = 26, 51$; $K = 1$. The flap rotates according to schedule 1. The rotation point is defined as $(x, y) = (0.6, 0.0)$ and is fixed in time. Segment 1 is a variable segment which connects the

forward part of the plate to the flap. As the flap rotates, the variable segment will keep its original tangent vector at the beginning of the segment (J=15) and maintain tangency to the flap at the end of the segment (J=26). Segment 3 is also a variable segment and represents the outflow boundary for the domain. Both end tangency conditions specify that the segment should maintain a normal to the adjacent segments. Thus while the flap rotates the outflow boundary will remain normal to the flap. The single schedule contains three data points and specifies that a rotation of 15 degrees will occur over a time interval of 1.0 time units. The schedule extends to a time of 5.0 time units to assure it is greater than `TFINAL`. Fig. 1 shows the grid for the inviscid computation of the flap when the flap is fully rotated.

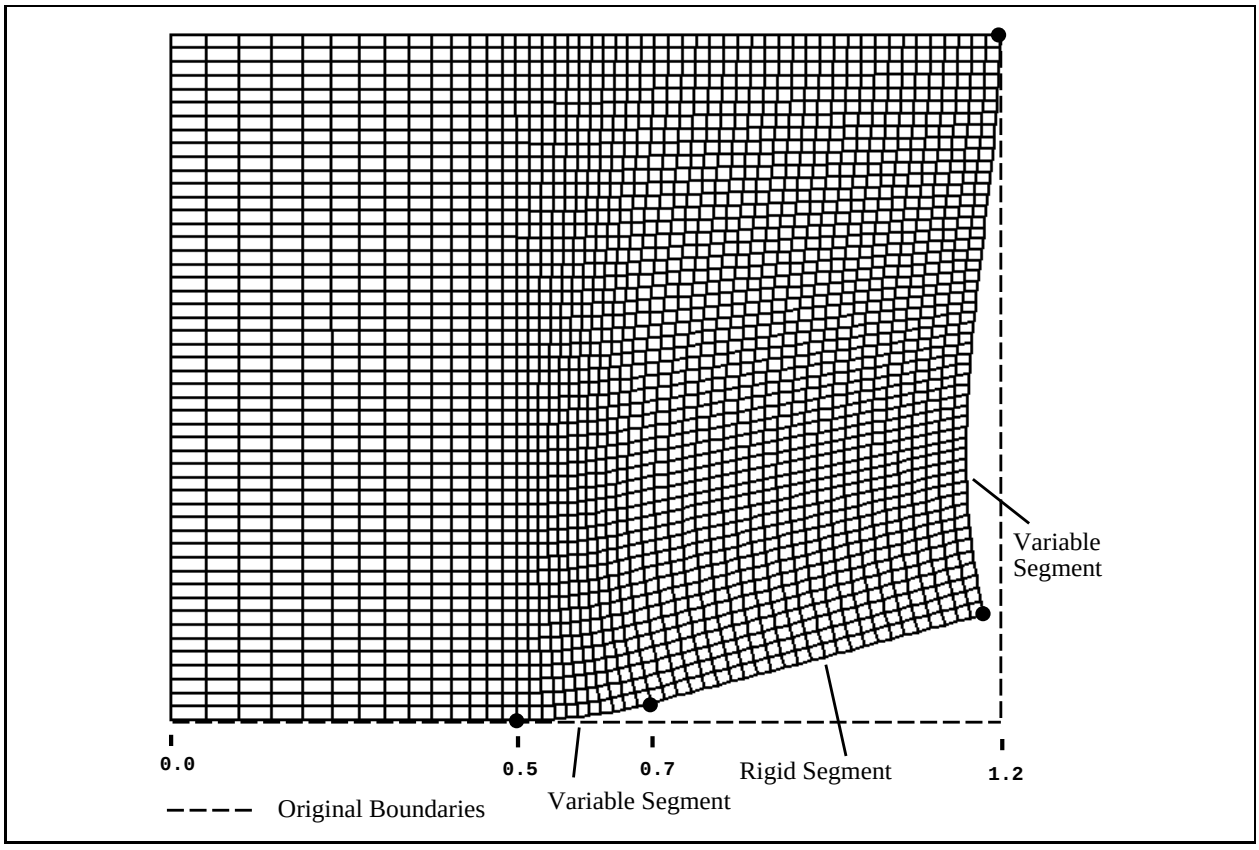

Figure 1: **The grid for the inviscid case for the flap at 15 degree rotation in Mach 2.5 flow.**

7.9 Dynamic Grid Include File (`dgmod.inc`)

The dynamic grid include file `dgmod.inc` contains several parameter statements that may need to be reset and NPARC v3.1 recompiled for large problems. The parameter `nmgmax` specifies the maximum number of dynamic segments and is currently set at 20. The parameter `nsfmax` specifies the maximum number of sub-faces (or dynamic segments) that can be placed in the j-coordinate direction and is currently 10. The parameter `nshmax` specifies the maximum number of schedules that are allowed and is currently set to 10. The parameter `nsdmax` specifies the maximum total number of data points that can be listed in all the schedules and is currently 50. If any of these values require changing, one should edit the file `dgmod.inc` and NPARC v3.1 should be recompiled.

7.10 Grid Regeneration

When `IGRDYN = 2` some grid regeneration is performed. NPARC v3.1 uses the grid read in from the restart file as the base grid. As a dynamic segment of the boundary grid moves, the grid in the region of that segment

is regenerated. The rest of the grid remains the same. This selective grid regeneration save computational effort. Implied in this approach is that the dynamic grid segments mainly exist along the constant-k surfaces. A block is then divided into sub-blocks containing the segment. The grid regeneration is then limited to the sub-block. The grid regeneration uses an algebraic transfinite interpolation method to efficiently regenerate the grid based on the updated boundary grid.

The situation may occur that the grid read in from the restart file is changed significantly by the grid regeneration at the initial time step.

If grid motion occurs at a block interface, one must specify the same grid dynamics for segments of both blocks at the interface in order to assure that the grid points remain contiguous.

7.11 Three-Dimensional Dynamic Grids

Three-dimensional dynamic grids assume that the regions with moving grids have a "quasi-2d" or axisymmetric nature, which implies that in the grid surfaces at constant-l coordinates are planar. The "quasi-2d" assumption requires that the grid at each l-coordinate have constant-z grid coordinates. The axisymmetric assumption requires that the grid at each l-coordinate have constant-θ grid coordinate where θ is the angle about the x-axis, which is assumed to be along the j coordinate. This assumption restricts the three-dimensional moving grid capability, but allows for the moving grids to be updated on a single plane with transformations to the remaining grid planes in the l coordinate direction. Thus three-dimensional flows may be computed with moving grids without a large computational cost attributed to the moving grids. The three-dimensional grid in regions of the grid that do not move are only limited by the requirements of NPARC v3.0. An additional requirement is the number of grid points in the l-coordinate direction should be greater than 2 in order for the dynamic grid information to be properly stored.

7.12 Advanced Grid Motion

If a more advanced grid motion capability is desired, one can replace the dynamic grid module (`dgmod.f`, `dgmod.inc`) The module interfaces NPARC v3.1 in `initia` through calls to subroutines `dginit`, `dgread`, `dgecho`, `dgset`, and `dgset0` and in `oneiter` with calls to subroutines `dgsch`, `dgrpt`, `dgnew`. The subroutine `dgmet` is attached to `dgmod.f` and would need to be put into a separate file or attached elsewhere.

8 Restart File

The focus of NPARC v3.1 is on solving unsteady flowfields with possible moving grids. This required expanding the restart file to possibly including two time levels of the solution and the grid speeds at the last time level. A new restart file format is available for NPARC v3.1 in which the first line contains the step count `NC`, the nondimensional time of the grid and solution `TRSTFL`, the time step between the old (QO) and current (QN) solutions `DTRSTF`, and flags `IRSTGS`, `IRSTQO`, and `IRSTTU`. The flags allow explicit determination of what is included in the restart file. They indicate:

```
IRSTGS  Flag to indicate if grid speeds are included in restart file
        (=0) Not included, (=1) Included
IRSTQO  Flag to indicate if old solution is included in restart file
        (=0) Not included, (=1) Included
IRSTTU  Flag to indicate if turbulence quantities are included in restart file
        (=0) Not included, (=1) Included
```

For Fortran code that reads in the new format of the restart file is the following:

```
   read (2) nc, trstfl, dtrstf, irstgs, irstqo, irsttu
   do  i = 1, nblock
     read (2) jmax, kmax, lmax
     read (2) (( x(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax ),
  &           (( y(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax ),
```

```
&           (( z(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax )
   if ( irstgs .eq. 1 ) then
     read (2) (( gx(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax ),
&             (( gy(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax ),
&             (( gz(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax )
   endif
   if ( irstqo .eq. 1 ) then
     read (2) (( qo(j,k,l,m), j=1,jmax), k=1,kmax), l=1,lmax), m=1,5 )
   endif
   read (2) (( qn(j,k,l,m), j=1,jmax), k=1,kmax), l=1,lmax), m=1,5 )
   if ( irsttu .eq. 1 ) then
     read (2) (( f1(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax),
&             (( f2(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax),
&             (( f3(j,k,l), j=1,jmax), k=1,kmax ), l=1,lmax)
   endif
 enddo
```

All the data written to the restart file is in non-dimensional unit. Thus `TRSTFL` and `DTRSTF` are in non-dimensional time units regardless of the value of `IREALT`.

NPARC v3.1 automatically detects the format of the restart file when the restart file is read. When the restart file is written, the new format will be used when `ISOLVE = 2` or `IGRDYN = 1, 2`. When `ISOLVE = 2`, the flag is set `IRSTQO = 1`. When `IGRDYN = 1, 2`, the flag is set `IRSTGS = 1`. One can force NPARC v3.1 to write the restart file in the new format through the use of the input variable `IRSTFL`. A value of `IRSTFL = 1` indicates that the new format is to be used. The default value is `IRSTFL = 0`, which will cause NPARC to write the restart file in the original format.

```
IRSTFL  Flag to indicate the format to write the restart file
        (=0) Original format (default)
        (=1) New format
```

9 New Options for Output

Several new output options have been added to NPARC v3.1 to facilitate the monitoring and output of time-varying computations.

Iteration Output The screen output containing the convergence information with respect to iterations now contains the physical time of the computation in both non-dimensional and dimensional time units.

Sequence of Grid and Solution Files Second, it is now possible to output grid and solution Plot3D files at a frequency of time steps. The input `NQFREQ` is an integer that controls the number of iterations between the writing of these files. A negative or zero value of `NQFREQ`, the default, indicates that no files are to be written. The files are written to unformatted files with names `u###x.dat` and `u###q.dat` where `###` is the value of an integer counter. The input `NFWRIT` indicates the starting value of the counter. The default is `NFWRIT = 1`; however, in a restart, one may set `NFWRIT` to a higher value to avoid overwriting files from a previous run. For example, specifying `NFWRIT = 47` would cause the first files written to be `u047x.dat` and `u047q.dat`. The value of `NFWRIT` must be greater than 0 and less than 1000. The counter is incremented by one after a set of files is written. The files are written at the start of the start of the computation to create files of the initial conditions. The grid files `u###x.dat` are only written if there is grid regeneration (i.e. `IGRDYN = 2`. The grid and solution files give snapshots of the grid at certain times and can be used for obtaining a variety of unsteady data. They can also be used in an animation of the unsteady flowfield. (*John Slater does have a simple animation program for SGI workstations, which can be obtained by contacting John at John.Slater@lerc.nasa.gov*). Another use of this capability is that it allows the monitoring of a developing solution (unsteady or convergence to steady-state). Also the files can be used to recover from a spurious solution by recreating a restart file at an intermediate point prior to the failure. This allows one to restart the computation with new parameters, but yet, starting with a more developed solution.

Tracking of the Time-Varying Shock Position For an unsteady flow problem involving the motion of a single normal shock wave, as found in supercritical inlets, the capability to output the streamwise shock location with time is now possible. The input variable `ISFREQ` controls the frequency of the time iterations at which to write the real time and shock position to the file **fort.85**. A negative value of `ISFREQ`, the default, indicates that no shock position information is to be written. The tracking algorithm searches along a J grid line, which is usually in the streamwise direction, and interpolates the x-coordinate at which the Mach number is 1.0. The input variable `INTBLK` specifies the block in which to track the shock. The input variables `KS` and `LS` define the `K` and `L` indices for the J grid line. For two-dimensional problems, `LS` is not needed. One can limit the extent of the J grid line by indicating the start of the line through the input variable `JSTRT`.

Output of the Time-Varying Static Pressure The input variable `IPFREQ` controls the frequency of the time iterations at which to write the static pressure. A negative value of `IPFREQ`, the default, indicates that information is to be written. Currently solutions with respect to time at three field positions can be written out. Specify the locations using (J1,K1), (J2,K2), and (J3,K3) for two-dimensions and (J1,K1,L1), (J2,K2,L1), and (J3,K3,L3) for three-dimensions.

Mass Flux The input variable `MFREQ` controls the frequency of time iterations at which to write the mass flux. A negative value of `MFREQ`, the default, indicates that information is to be written. The flux is checked along the streamwise direction specified through `JM1`, `JM2`, and `JM3`.

10 Boundary Conditions

NPARC v3.1 includes the Chung-Cole compressor face boundary condition [4]. The capability to vary in time the compressor-face Mach number and supersonic inflow conditions was also added. Some of the modifications to existing boundary conditions that account for dynamic grids are also discussed.

10.1 Chung-Cole Compressor Face Boundary Condition

The compressor face boundary condition is a subsonic outflow boundary. Thus, one physical and three numerical boundary conditions must be specified. The three numerical boundary conditions can be extrapolations of density and momentum components from the interior or other methods as described above. Several options exist for updating the energy at the boundary. One option for the physical boundary condition is to specify the static pressure. This option has not been favorable for internal flows because the boundary condition is reflective. When the compressor face is relatively close to the throat, such as in some high-speed inlets, the reflection of waves from the compressor face station will inhibit convergence of the flow to a steady-state. Thus it will take longer for a normal shock to develop in the diffuser.

The Chung-Cole compressor face boundary condition [4] is an alternative and is related to the Boeing compressor face boundary condition in NPARC v3.0 (TYPE 97) developed by Mayer and Paynter [9]. These boundary conditions are based on the observation from experimental tests that the Mach number remains fairly uniform at the compressor face, which corresponds to a fairly uniform mass flux through the compressor face. An average Mach number at the compressor face M_{cf} can be specified. The total pressure at the boundary points, $p_t(r)$, can be computed at the current time. The static pressure $p(r)$ can then be computed from the isentropic relation

$$p(r) \;=\; p_t(r) \left(1 + \frac{\gamma - 1}{\gamma} \; M_{cf}^2(r) \right)^{\frac{-\gamma}{\gamma - 1}} . \tag{27}$$

The new pressure is used to update the total energy at the boundary.

The local compressor face Mach number $M_{cf}(r)$ can vary, as long as, the average Mach number is equal to that specified. For an inviscid flow, one can set the local compressor face Mach number equal to the specified average compressor face Mach number

$$M_{cf}(r) \;=\; M_{cf} . \tag{28}$$

For viscous flows, the local compressor face Mach number will vary with the boundary layer. One way to specify the local Mach number so that it varies with the boundary layer is

$$M_{cf}(r) \;=\; M_{cf}(r)\frac{M_{cf}}{M^*_{cf}} \tag{29}$$

where M^*_{cf} is the mass-averaged compressor face Mach number as computed at the current time. Thus if the computed average compressor face Mach number is lower than the specified value, then the above relation increases the local Mach number.

This boundary condition is implemented when `TYPE = 4` in the namelist `BOUNDS`.

10.2 Perturbations at Downstream and Upstream Boundaries

One feature of NPARC v3.0 is that a sequence can be specified for changing the character of the time step according to the iteration count. A non-zero value of `NUMDT` indicates the number of time step sequences, which are specified by the values of `DTCAP` in the arrays `DTSEQ(I)` and `ITERDT(I)` through the namelist `SEQDT`.

More options were added in NPARC v3.1 to the existing time sequencing to allow time-accurate perturbations at either at the compressor face or the supersonic inflow boundaries. The time and perturbations sequence is now controlled by two input variables `NUMDT` and `NPTSEQ`. The `NPTSEQ` is the number of perturbation segments as defined in the `SEQDT` namelist block as discussed below. Currently the following four combinations are possible:

`NUMDT =< 0` and `NPTSEQ =< 0`. This combination means that there is no time step size sequencing (fixed time step or CFL number) and no perturbation of the input data at the boundaries. This combination is the default.

`NUMDT > 0` and `NPTSEQ =< 0`. This combination means that the time step size varies sequentially. If `IREALT = 0`, then `DTSEQ` contains the CFL number. If `IREALT = 1` then `DTSEQ` contains the dimensional time step. The `ITERDT` array contains the number of iterations for which to maintain the value of `DTSEQ`. An example is

```
$SEQDT
   DTSEQ(1)=1.0E-5, ITERDT(1)=10,
   DTSEQ(2)=2.0E-5, ITERDT(2)=20,
   DTSEQ(3)=5.0E-5, ITERDT(3)=20,
   DTSEQ(4)=1.0E-4, ITERDT(4)=250,
$END
```

`NUMDT > 0` and `NPTSEQ > 0`. This combination is not possible currently because only a fixed time step size is allowed for the convenience of tracking the total elapsed time accurately for perturbation cases.

`NUMDT =< 0` and `NPTSEQ > 0`. This combination means that only perturbations of the boundary conditions exist. WARNING: `NC` should be set to '-2' in the namelist if `INTCF = 1` or `INTUP = 1` (when interpolation is desired). Refer to the next two sections for the usage of `INTCF` and `INTUP`. This will also set the dimensional time count to 0.0. The time step should be fixed. Set `IREALT = 1` and use `REALDT` as the dimensional time step size. Dimensional time is specified through `TDIST(I)` where 'I' stands for the iteration sequence. The values of `PN(I)`, `TN(I)`, and `XMN(I)` are new values of pressure, temperature, and Mach number, respectively, at the inflow boundary and `CFMN(I)` is the new compressor face Mach number at `TDIST(I)`. These values represent the ratios between before the perturbation and after it. In the following example, the namelist `SEQDT` input data, `TN(1) = 1.02` means that the new temperature at time = 5.0e-4 seconds is a 2% increase over the original value. It is assumed that the perturbation distribution starts at a time of 0.0 seconds and perturbation values of 1.0.

```
$SEQDT
   TDIST(1)=5.0E-4, PN(1)=1.00, XMN(1)=1.00, TN(1)=1.02, CFMN(1)=1.00,
```

```
   TDIST(2)=1.0E-3, PN(2)=1.00, XMN(2)=1.00, TN(2)=1.03, CFMN(2)=0.98,
   TDIST(3)=1.5E-3, PN(3)=1.00, XMN(3)=1.00, TN(3)=1.04, CFMN(3)=0.97,
   TDIST(4)=2.0E-3, PN(4)=1.00, XMN(4)=1.00, TN(4)=1.02, CFMN(4)=1.00,
   TDIST(5)=2.5E-3, PN(5)=1.00, XMN(5)=1.00, TN(5)=1.00, CFMN(5)=1.00,
$END
```

10.3 Time Variation of the Compressor Face Mach Number

When `NUMDT =< 0` and `NPTSEQ > 0`, the perturbations at the compressor face can be imposed in three different ways. A new input variable `INTCF` (with a default `INTCF=-1`) is used to differentiate these cases. Another input variable `ICVOL` (with a default of `ICVOL=0`) can be used to specify the assumption of constant volumetric flow (`ICVOL=1`). The three combinations are:

`NPTSEQ > 0` and `INTCF = -1`. For this combination, there is no perturbation sequence with respect to time. The input variable `CFMACH` (mass-averaged compressor face Mach number) is used directly and the change of this value represents a step change. The input variable `INTCF = -1` should be used when there is a sequential perturbation at the upstream boundary, but at the compressor face, no such sequencing is desired.

`NPTSEQ > 0` and `INTCF = 1`. For this combination, linear interpolations can be performed when one does not know the boundary values(BVs) at intermediate time steps. Note again, `IREALT` should be set to 1 and the dimensional time step size `REALDT` should be used for the simulation. For example, if you know the BVs' at t = 0.01, 0.02, 0.025 seconds, respectively, but your time step size is smaller such as dt = 0.002 second (fixed value), the code needs the boundary values at 0.004,0.006, 0.008 ...etc which are interpolated from the given BVs. ('step change' of BV is often enough for certain problems to avoid using the interpolation).

`NPTSEQ > 0` and `INTCF = 0`. For this combination, the ratio of the boundary values between perturbations can be specified using `CFMN(NPERT)`. For example, `CFMN(I) = 0.98` means that the compressor face Mach number has decreased by 2% at time `TDIST(I)`. The `CFMMN(NPERT)` is used where, `CFMMN(NPERT) = CFMACH * CFMN(NPERT)`. Again, for viscous flow cases, the Mach numbers are scaled by `FNMACH(K) / FMAVG`.

10.4 Time Variation of the Supersonic Inflow Boundary

When in flight, supersonic aircraft encounter pressure, temperature and velocity disturbances from a variety of sources. If the propulsion system uses a mixed-compression inlet, these phenomena can cause undesirable inlet unstart which could result in a severe propulsive inefficiency or difficulty in control. Time accurate analysis of unstart tolerance is essential to help understand the aircraft control.

To handle these phenomena, a new boundary condition allowing the perturbation of properties at the supersonic inflow boundary has been added and specified through `BCTYPE = 5`. Mach number, pressure, or temperature can be perturbed at this boundary with respect to time and each case is selected by the input variable `ICASE`.

```
ICASE  Inflow perturbation property
       (1) Mach number
       (2) pressure
       (3) temperature
```

The input variable `INTUP` (default = 0), controls the interpolation at the boundary.

When the Mach number changes, the free-stream velocity changes but the pressure and density are fixed. In case of a pressure change, the static temperature is fixed while the density varies. When the temperature changes, the static pressure is fixed while the density varies.

Three cases are possible:

`NPTSEQ < 0`. Only step changes in pressure, temperature, or Mach number are allowed with no time sequencing. New values are read in as input data, `PNEW`, `TNEW`, or `XMNEW` in the namelist `INPUTS`.

`NPTSEQ > 0` and `INTUP = 0`. The temperature, pressure, and Mach varies stepwise with respect to time.

`NPTSEQ > 0` and `INTUP = 1`. A linear interpolation is performed.

10.5 Control Flag Ranges and Stepwise Changes for a Perturbation Study

The ranges for input variables controlling perturbation are

```
irealt = {0,1}
icvol  = {0,1}
icase  = {1,2,3}
intcf  = {-1,0,1}
intup  = {0,1}
```

To set up the parameters for a step change occurring only once at the compressor face or the supersonic inflow boundaries (when `NPTSEQ < 0`):

(a) Compressor face Mach step changes but no upstream perturbation:

```
at the compressor face boundary, choose BCTYPE = 4,
at the upstream boundary, BCTYPE = -10 (supersonic inflow, no change)
CFMACH: new value specified in the namelist input
```

(b) Upstream perturbation, compressor face Mach number does not change:

```
at the compressor face boundary, choose BCTYPE = 4
at the upstream boundary, BCTYPE = 5
CFMACH: fixed
   ICASE = 1 : read in new Mach number through XMNEW
   ICASE = 2 : read in new pressure through PNEW
   ICASE = 3 : read in new temperature through TNEW
```

10.6 Solid Wall Boundary Conditions for Dynamic Grids

The boundary conditions are the primary mechanism by which the flow simulation realizes that a boundary is in motion. This required modification of several boundary condition routines. These are discussed below.

The physical boundary condition for a solid, slip wall is

$$\rho_B \left(\vec{V}_B - \vec{g}_B \right) \cdot \hat{n} = 0 \tag{30}$$

where $\vec{g}_B$ is the velocity of the boundary

$$\vec{g}_B = (x_\tau)_B \ \hat{\imath} + (y_\tau)_B \ \hat{\jmath} + (z_\tau)_B \ \hat{k}. \tag{31}$$

For a viscous wall, the additional boundary condition (no-slip) is

$$\rho_B \left(\vec{V}_B - \vec{g}_B \right) \cdot \hat{t} = 0 \tag{32}$$

where $\hat{t}$ is the tanget vector at the wall.

11 Miscellaneous Modifications

Various minor changes have been made to the code since the 3.0 release, these include:

- Solution monitoring of stagnation pressure and temperature. Available by setting the `BOUNDS` namelist variable `TYPE` to xx02 for pressure and xx04 for temperature.
- Removal of the restriction on type 70 and type 77 boundary conditions where indices must run in the same direction. By setting the `JEDGE`, `KEDGE`, or `LEDGE` variable for a boundary the index direction may be reversed. For the 2D code, `xEDGE=-1` reverses the index direction. For the 3D code, `xEDGE=-1` or `xEDGE=-3` reverses the direction of the first index while `xEDGE=-2` or `xEDGE=-3` reverses the direction of the second index.

- Namelist input may begin in column 1. By default, namelist input begins in column 2. This allows for an *echo* feature by entering 'E' in column 1. By entering the namelist group identifier (i.e. `$INPUTS`) in column 1, the echo feature is disabled and namelist data may begin in column 1 of subsequent lines. The namelist column selection is reset to 2 before scanning each namelist group.
- Simplified serial implementation. Version 3.1 restores some of the simplicity of the 2.0 code for the serial case. This is done by replacing the functionality of the files `main.f`, `master.f`, and `worker.f` with the new file `nparc.f`.
- Dynamic allocation of interface and interpolation storage. This is controlled by the `INPUTS` namelist variables `MIFACE` and `MINTERP` which take the place of the compile-time parameters of the same name.
- Code coupling via VCE version 2.4. VCE (Visual Computing Environment) is a cooperative effort between CFD Research Corporation and NASA LeRC. Supporting VCE has required some restructuring of the code.
- A backup copy of the restart file can be retained by setting `INPUTS` namelist variable `IBACKUP=1`. When storage space permits, this is useful to avoid losing data in the case of a write error when updating the restart file during a calculation.

12 Test Cases

The following test cases demonstrate the input procedures, performance, and accuracy for unsteady flow simulations with and without dynamic grids. The files for these test cases are found in the **tests** directory and will be executed when a `make` command is executed in the respective directories.

12.1 Static 15 Degree Wedge in Mach 2.5 Flow

This simple test case involves a 15 degree wedge in Mach 2.5 flow with the assumption of inviscid flow. The flow properties behind the attached oblique shock can be exactly computed from compressible flow theory if perfect gas is assumed. The files associated with the two-dimensional test case are located in the directory **tests/slater2d/swedge**. A single block restart file is generated with the program `wedic.f`, which reads in the grid file `wedx.fmt` with a grid size of 55 x 25. A three-dimensional test case is located in the directory **tests/slater3d/swedge** and is a simple extrusion of the two-dimensional case with 5 grid points in the k-direction. The NPARC input file is `wedge.in`. The Euler implicit method (`ISOLVE=1`) is used for 500 time steps with local time-stepping to obtain the steady-state solution. A CFL number of `DTCAP` = 5.0 is used. The computation is executed as

```
wedic
nparc#ds < wedge.in > wedge.out
```

where `#` corresponds to the nparc2ds or nparc3ds code.

The files `wedge.out` contains the NPARC output for the run. The computation prints out the solution along the k=1 line. The computed solution results at j=55, k=1 are

	Theory	2D	Error(%)	3D	Error(%)
Mach Number:	1.8735	1.8996	(1.39%)	1.8957	(1.18%)
Pressure ratio:	2.4675	2.4695	(0.08%)	2.4696	(0.08%)
Temperature ratio:	1.3220	1.3109	(-0.84%)	1.3129	(-0.84%)

12.2 Flying 15 Degree Wedge in Mach 2.5 Flow

This test case solves the above problem, except now the wedge is specified to move at a velocity of Mach 2.5 in a static flow field. The motion of the wedge is specified by imposing a velocity to the the grid of a constant, uniform velocity of Mach 2.5 to the left. This is done by setting the grid speeds in the restart file through the program `fwedic.f` which reads the grid file `wedx.fmt`. The files associated with this test

case are located in the directories **tests/slater2d/fwedge** and **tests/slater3d/fwedge.** The input file is `fwedge.in`, which contains the same parameters as the previous problem except now the restart file flag is set to `IRSTFL = 1` to indicate that the new restart format is used. Further, the input file has the input `IGRDYN = 1` to indicate that constant grid speeds are specified. The computation is executed as

```
fwedic
nparc#ds < fwedge.in > fwedge.out
```

The computed solution results at j=55, k=1 are

	Theory	2D	Error(%)	3D	Error(%)
Mach Number:	0.6265	0.6027	(-3.80%)	0.6031	(-3.74%)
Pressure ratio:	2.4675	2.4695	(0.08%)	2.4696	(0.08%)
Temperature ratio:	1.3220	1.3109	(-0.84%)	1.3129	(-0.69%)

The results are almost identical to the static wedge case (the outflow Mach number is slightly different). The plot of surface pressures with respect to x is identical for both cases. The plot of surface Mach numbers shows significant difference at the foot of the shock due to the opposite manner in which velocities are imposed (one due to freestream flow and the other due to the motion of the wedge). This is shown in Fig. 2. The file `fwedge.out` contains the NPARC output for the run. This case can also be run using the Runge-Kutta explicit method (`ISOLVE=5`) with `DTCAP = 2.0` with essentially the same results. This input file is `fwedge2.in`.

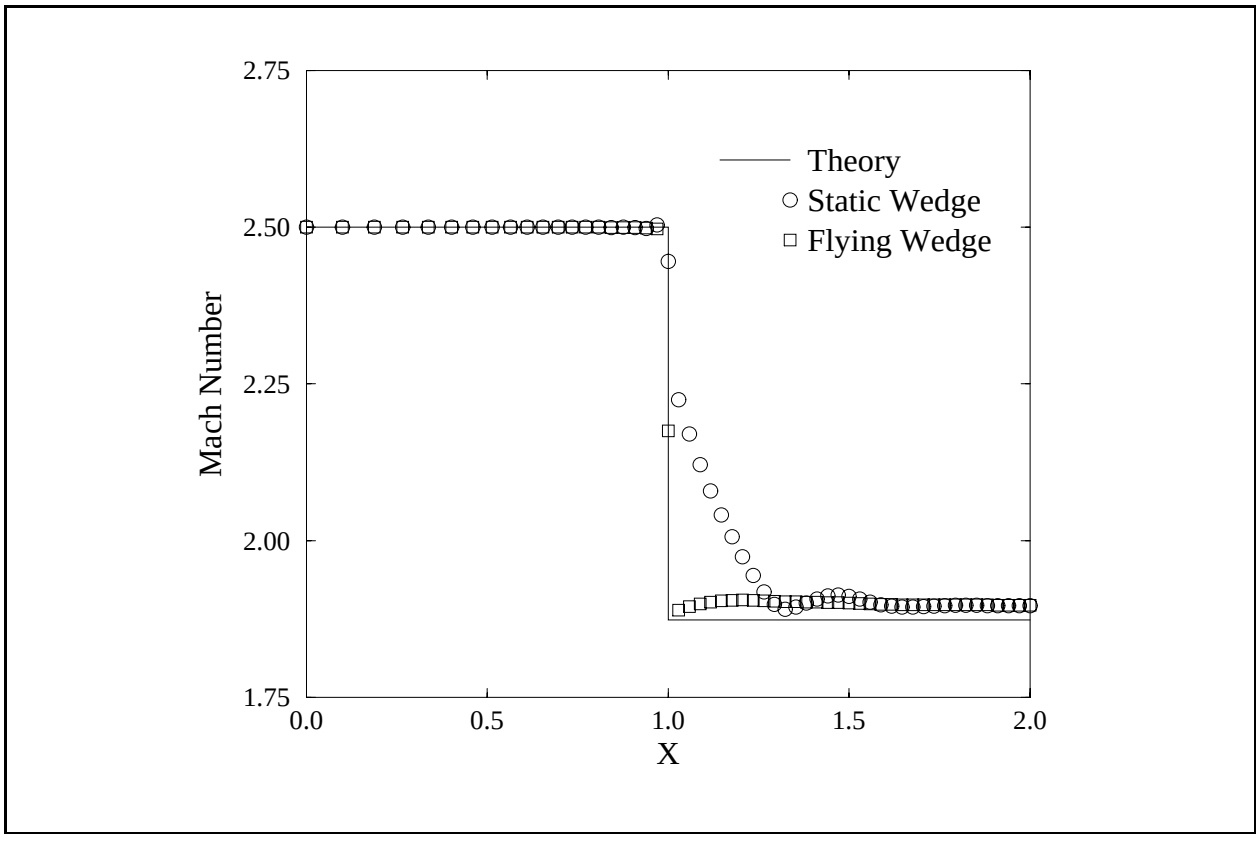

Figure 2: **The variation of the Mach number along the surface of a fifteen degree wedge in Mach 2.5 flow.**

12.3 Sod's Shocktube

Sod's shocktube problem is a test of the time-accuracy of the computational methods and involves a 10/1 pressure ratio across the diaphragm. The test case examines the flow after 0.20 seconds from the burst of the diaphragm. The analytic solution can be determined from perfect gas compressible flow theory for moving waves. The nondimensional pressure, density, Mach number, and velocity variations at $t = 0.02$ seconds as a function of length along the tube are available in the files `sodstp.ex`, `sodstr.ex`, `sodstm.ex`, `sodstu.ex`, respectively. The files associated with the axisymmetric test case are located in the directory **tests/slater2d/sod**. The files associated with the three-dimensional test case are located in the directory **tests/slater3d/sod**.

The restart file is generated with the program `sodic.f`. A uniformly-spaced grid with axial and transverse grid density of 100 and 21 grid points, respectively is generated. For the three-dimensional problem, 5 grid points are placed circumferentially with spacing of 2.5 degrees. The conditions are nondimensionalized with a reference pressure and density of unity. Using the perfect gas equation of state, the reference velocity (speed of sound) is the square root of the ratio of specific heats. The reference length is unity, therefore, the reference time is $t_{ref} = 0.845155$ seconds.

A series of three computations are performed with varying setting for `ISOLVE`.

```
   File       ISOLVE
 sodst1.in       1
 sodst2.in       2     (NSUBMX=3, TOLSUB=1.0E-12)
 sodst3.in       5
```

The computations are performed for a physical duration of 0.2 second, which corresponds to a nondimensional time of 0.236643 time units. For 500 time steps, this results in a constant, uniform nondimensional time step of 4.732863E-04 time units. The flow conditions at the final time along the centerline of the shocktube are output. The files of the form `sodst#.out`, where `#` is the run number, contain the NPARC output for each run. The files `sodstr#.do`, `sodstm#.do`, and `sodstp#.do`, where `#` is the run number, contain the density, Mach number, and pressure variations for each run. The computations are executed as

```
sodic
nparc#ds < sodst1.in > sodst1.out
nparc#ds < sodst2.in > sodst2.out
nparc#ds < sodst3.in > sodst3.out
```

Fig. 3 shows the variation of the density in the shocktube. There is considerable oscillation in the flow and the Newton subiteration method seems to improve the time accuracy over the backwards Euler method. The solution from the explicit Runge-Kutta method is coincident with the Newton iterative method.

12.4 Standing Shock Wave

An unsteady flow to test the time accuracy of a computation involves a Mach 1.3327 supersonic flow in a duct with an initially standing (stationary) normal shock wave in which the Mach number at the exit of the duct is 0.7699 [10]. A perturbation of 5% of the exit Mach number forces the shock to move forward in the duct. The duct extends from x=-10.0 inches to x=10.0 inches with the normal shock initially placed at x=0.0 inches. The perturbation occurs at a time of 0.001264 seconds and flow conditions are examined at a time of 0.025 seconds. The inflow static temperature is 402.9004 Rankine. An analytic solution is possible for the position of the shock with time and the file `standxs.ex` contains this variation. The file `standmh.ex` contains the time variation of the Mach number at a location x=-5.0 inches. The files associated with the axisymmetric test case are located in the directory **tests/slater2d/stand**. The files associated with the three-dimensional test case are located in the directory **tests/slater3d/stand**.

For the NPARC computations, the program `standic` generates the restart file with the initial conditions, which contain the stationary normal shock. The inflow temperature and a length scale of 1.0 ft correspond to a reference time of 1.016293E-03 seconds using the perfect gas assumption (the value of the gas constant is 1716.47). The NPARC computations start at the time of the perturbations ($t = 0.001264$ sec), which corresponds to a nondimensional time of 1.243736 time units. The NPARC computation proceeds until the

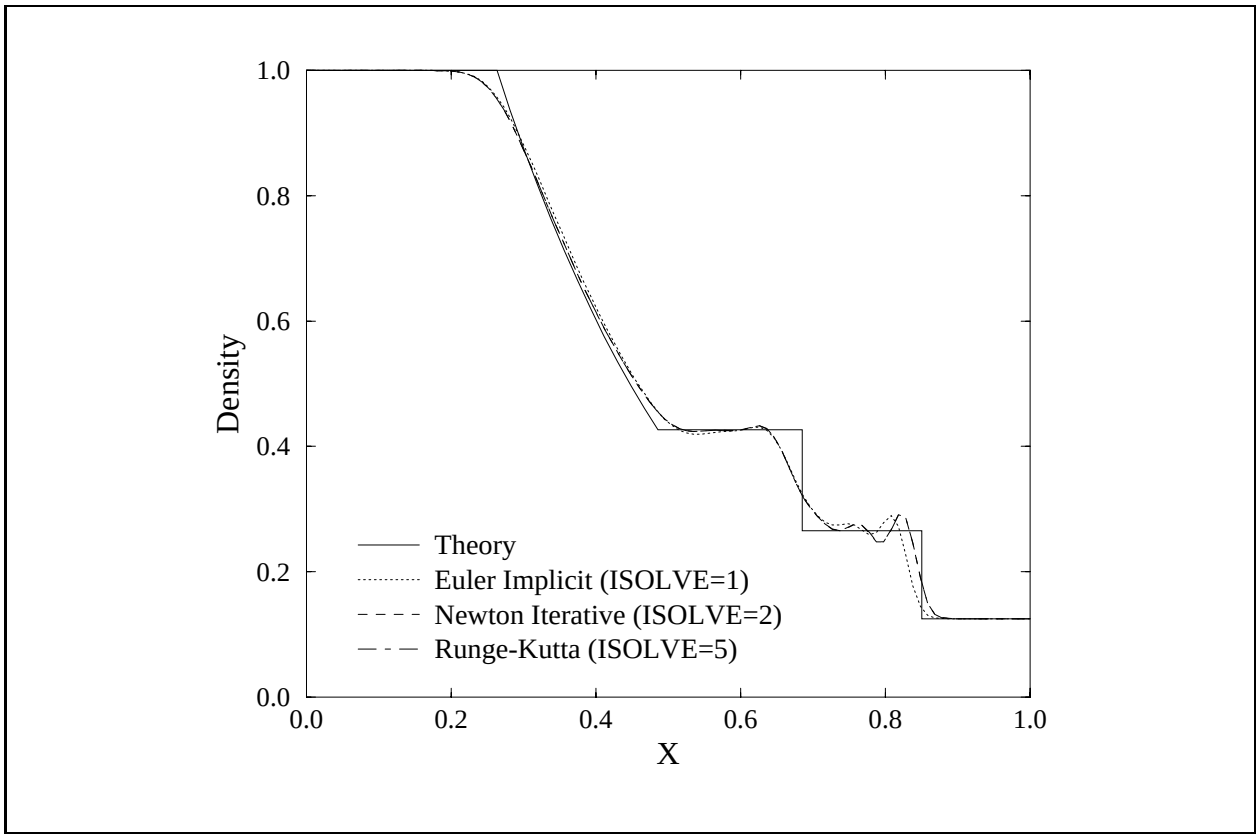

Figure 3: **The variation of density in the shock tube at $t = 0.2$ seconds.**

final time of $t = 0.019378$ seconds, which corresponds to the time at which the shock should be positioned at $x =$ - 5.0 inches and is a nondimensional time of 19.067607 time units. The NPARC input files are of the form `stand#.in` where `#` corresponds to the run number. The computations are executed as

```
sodic
nparc#ds < stand1.in > stand1.out
nparc#ds < stand2.in > stand2.out
nparc#ds < stand3.in > stand3.out
nparc#ds < stand4.in > stand4.out
nparc#ds < stand5.in > stand5.out
```

The NPARC computations output the conditions along the duct at the final time. The files of the form `stand#.out`, where `#` is the run number, contain the NPARC output for each run. The position of the shock can be computed as the location at which the Mach number is unity. This location can be compared to the desired location of $x =$ - 5.0 inches. The results from the axisymmetric computation are

Run	ISOLVE	IVARDT	DTCAP	NSTEPS	Xshock	%Error
1	1	0	0.01782387	1000	-4.276429	-14.47
2	2	0	0.01782387	1000	-5.048015	0.96
3	5	0	0.005	3565	-5.049640	0.99
4	2	5	5.0	1004	-5.047007	0.94
5	2	5	5.0	1004	-5.056984	1.14

The results from the three-dimensional computation are

Run	ISOLVE	IVARDT	DTCAP	NSTEPS	Xshock	%Error
1	1	5	5.0	4362	-3.563259	-28.73

2	2	0	0.004	4457	-5.058926	1.18
3	5	5	2.0	10903	-5.047745	0.95
4	2	5	5.0	4363	-5.053675	1.07
5	2	5	5.0	4363	-5.065694	1.31

The shock position data presented above shows that using the Newton subiteration method or the explicit method resulted in very good agreement with theory. The poor performance of the Euler implicit method justifies the greater computational expense of the Newton subiteration method. The fifth run uses the Chung-Cole compressor face boundary condition (TYPE=4), whereas runs 1-4 use the Boeing compressor face boundary condition (TYPE=97).

12.5 Rotating Flap

A simple case involving the relative motion of a segment of the boundary grid is the motion of a flap on a flat plate from a deflection from 0 to 15 degrees in Mach 2.5 inviscid flow. The initial conditions are specified as a pressure of 101 kPa and a temperature of 290 K.

The program `flapic.f` reads in the initial grid from the Plot3D file `flap.fmt` which has 51 streamwise and 52 transverse grid points. The plate has a length of 1.2 meters. The flap is located at the last 0.5 meters of the plate. Fig. 1 shows the grid at the full rotation of the flap. The file `flap.in` is the input data file for NPARC. The flap deflection is specified to occur over a non-dimensional time of 3.0 time units. The Newton iterative method (`ISOLVE=2`) is used with variable time stepping (`IVARDT=5`) and a CFL number of `DTCAP` = 1.0. The computation is performed assuming a two-dimensional flow. The files associated with this test case are located in the directories **tests/slater2d/flap** and **tests/slater3d/flap**.

The motion of the flap and the deformation of the grid is indicated by the grid dynamics flag, `IGRDYN = 2`. The file `flap.54` contains the dynamic grid parameters. This file is copied to `fort.54` prior to the computation. The parameters indicate that the flap is the segment from j = 26 to 51 at the k = 1 boundary. The center of rotation is specified at (x, y) of $(0.6, 0.0)$. The segments from j = 15 to 26 at k = 1, and from k = 1 to 52 at j = 51 are variable segments allowed to deform according to the flap motion. The flap rotates through 15 degrees over a non-dimensional time of 1.0 time units and then remains fixed at that deflection for the remainder of the computation.

The computation is executed as

```
flapic
cp flap.54 fort.54
nparc#ds < flap.in > flap.out
```

The file `flap.out` presents the output from NPARC. The properties along the flap surface are printed out at the final time. At the final time, the steady-state solution is obtained and should be the same as the flow solution of the wedge problem presented above. The results for the flow behind the shock as measured at j=55, k=1 are:

	Theory	2D	Error(%)	3D	Error(%)
Mach Number:	1.8735	1.9097	(1.93%)	1.9072	(1.80%)
Pressure ratio:	2.4675	2.4712	(0.15%)	2.4704	(0.12%)
Temperature ratio:	1.3220	1.3037	(-1.38%)	1.3052	(-1.27%)

The problem was simulated assuming viscous, turbulent flow using the Baldwin-Lomax turbulence model. There are now 80 grid points in the transverse direction. The files associated with this test case are located in the directories **tests/slater2d/vflap** and **tests/slater3d/vflap**. The results for the flow behind the shock as measured at j=55, k=1 are:

	Theory	2D	Error(%)	3D	Error(%)
Pressure ratio:	2.4675	2.4911	(0.96%)	2.4919	(0.99%)

12.6 Piston Expansion

A simple test case involving a deforming boundary is a straight duct 10 meters long with its right boundary moving to the right at a constant speed of 100 m/sec, which may simulate the motion of a piston. The initial conditions are specified as a pressure of 101 kPa and a temperature of 293 K. The velocities are initially zero. A centered expansion wave is formed at the piston surface and propagates into the duct. The method of characteristics provides for an analytic solution for the flow properties with respect to space and time[2]. The file `pistonr.ex` contains the axial variation of density at the final time of $t_f = 0.01$ seconds. The files associated with this test case are located in the directories **tests/slater2d/piston** and **tests/slater3d/piston**.

The program `pistonic.f` creates the initial grid and solution for NPARC. A reference length of 10.0 meters is used. The uniformly-spaced grid has 202 axial and 12 radial grid points. The file `piston.in` is the input data file for NPARC. The final time of $t_f = 0.01$ seconds corresponds to a nondimensional final time of `TFINAL` = 0.3431143 time units. The Newton iterative method (`ISOLVE=2`) is used with variable time stepping (`IVARDT=5`) and a CFL number of `DTCAP` = 0.5. The computation is performed assuming an axisymmetric flow.

The motion of the right boundary and the deformation of the grid is indicated by the grid dynamics flag, `IGRDYN = 2`. The file `piston.54` contains the dynamic grid parameters. This file is copied to `fort.54` prior to the computation. The parameters indicate that the right boundary translates 0.1 length units (1.0 meters) at a constant rate over the time interval.

The computation is executed as

```
pistonic
cp piston.54 fort.54
nparc#ds < piston.in > piston.out
```

The file `piston.out` presents the output from NPARC. The properties along the center of the duct are printed out at the final time and the file `pistonr.do` which contains the density along the duct at the final time is extracted. Fig. 4 presents a comparison of the spatial variation in density in the duct at the final time. The NPARC solution was slightly more dissipative and introduced oscillations at the tail of the expansion.

12.7 Rapid Collapse of a Bump in a Duct

This case involves the rapid collapse of a flexible bump in an annulus[6],[14]. The objective of this case was the generation of an individual, well-characterized, short-duration acoustic pulse of amplitude of 10% of the ambient pressure. The case involves an axisymmetric bump formed on the hub of a constant-area annulus. The bump collapses within 0.8 milli-seconds (msec) to form a cylindrical section flush with the hub. Two expansion pulses are formed - each traveling axially at the speed of sound away from the bump. As they travel along the duct, they form a planar structure. The duct is 127.0 centimeters long with a hub radius of 5.70 centimeters and a case radius of 7.57 centimeters. The bump has a length of 9.5 cm and starts at a distance 81.6 centimeters from the left end of the duct. The bump is formed by the deformation of a thin-walled silicon rubber (RTV) tube segment (called the "boot"). The deformation is due to the pressurization of the driver section of a shock tube within the hub. The collapse of the bump occurs when the diaphragm of the shock tube is burst by the actuation of a spear mechanism. The test case considers a bump height h to duct height H of $h / H = 0.50$. The bump collapse occurs in initially stagnant conditions (no flow) with a static pressure of 101 kPa and temperature of 293 Kelvin. Thus, the induced flow and generated pulses are due entirely to the bump motion. The file `duct50p.ex` contains the pressure data from the experiment for the sensor located at 40.64 centimeters from the left end of the duct, which is about 40.96 centimeters from the bump. The files associated with this test case are located in the directories **tests/slater2d/duct** and **tests/slater3d/duct**.

The program `duct50ic.f` reads in the initial grid from the file `duct50.fmt` and creates the initial solution for NPARC. The grid has dimensions of (719,12) with uniform spacing in the transverse direction (from hub to case). The radius of the case is used as the reference length.

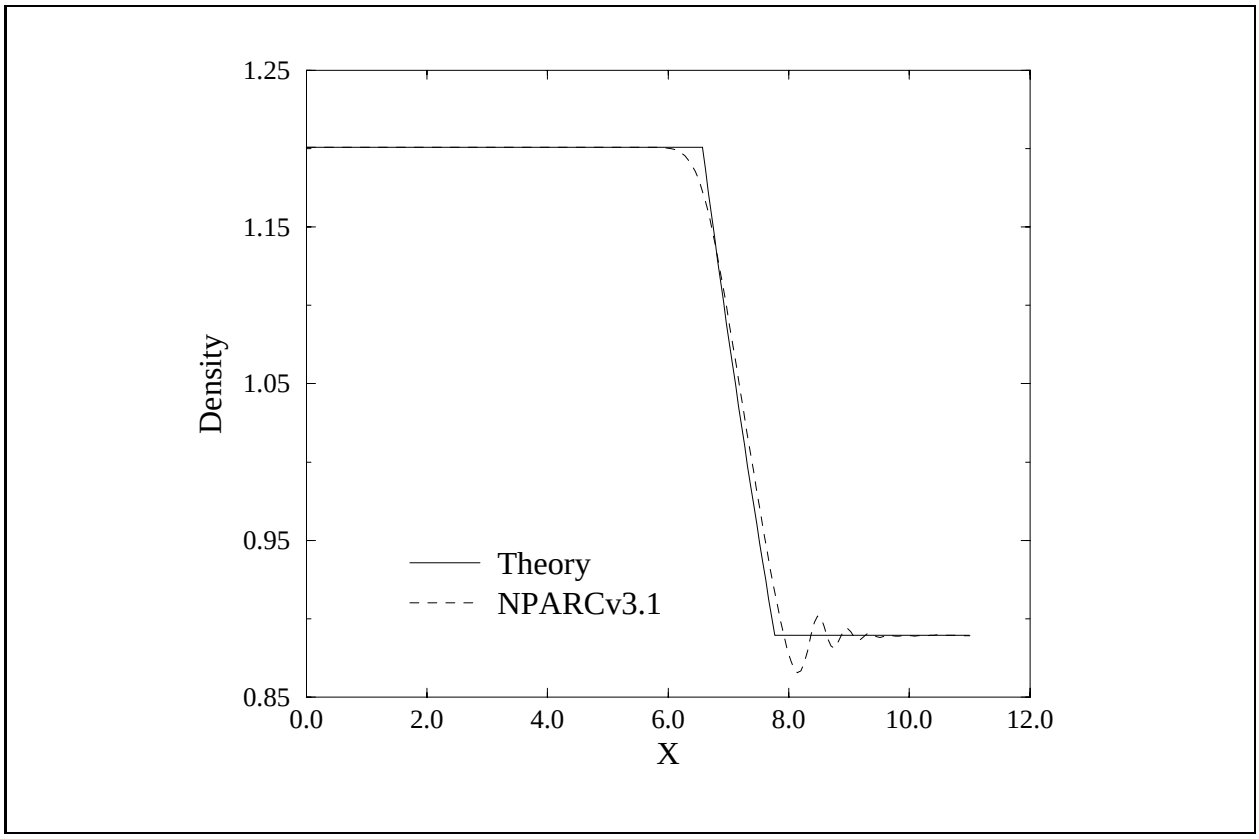

Figure 4: **The spatial variation of density through a centered expansion wave generated by a rapidly moving piston.**

The file `duct50.in` is the input data file for NPARC. The final time of $t_f = 0.003$ seconds corresponds to a nondimensional final time of `TFINAL` = 13.60 time units. (Note that the bump collapses in a nondimensional time of 3.627034 time units.) The Newton iterative method (`ISOLVE=2`) is used with variable time stepping (`IVARDT=5`) and a CFL number of `DTCAP` = 0.2. The computation is performed assuming an axisymmetric flow.

The collapse of the bump and the deformation of the grid is indicated by the grid dynamics flag, `IGRDYN = 2`. Fig. 5 shows the sequence of grid deformation as the bump collapses. The file `duct50.54` contains the dynamic grid parameters. This file is copied to `fort.54` prior to the computation. The parameters indicate that the bump is located from j = 461 to 517 at the k = 1 boundary and that its type is a deforming based on the coding in the `deform` subroutine of the `dgmod.f` file.

The computation is executed as

```
duct50ic
cp duct50.54 fort.54
nparc#ds < duct50.in > duct50.out
```

The file `duct50.out` presents the output from NPARC. The properties along the center of the duct are printed out at the final time and the file `duct50p.do` which contains the pressure at the sensor location.

Fig. 6 shows the comparison of the time-variation of the static pressures recorded in the experiment at a sensor location with those computed using NPARC. After the main pulse has passed, a smaller, decaying oscillation is noticed, which is believed due to the drum-like vibration of the boot and the possible rebounding of the boot from the hub.

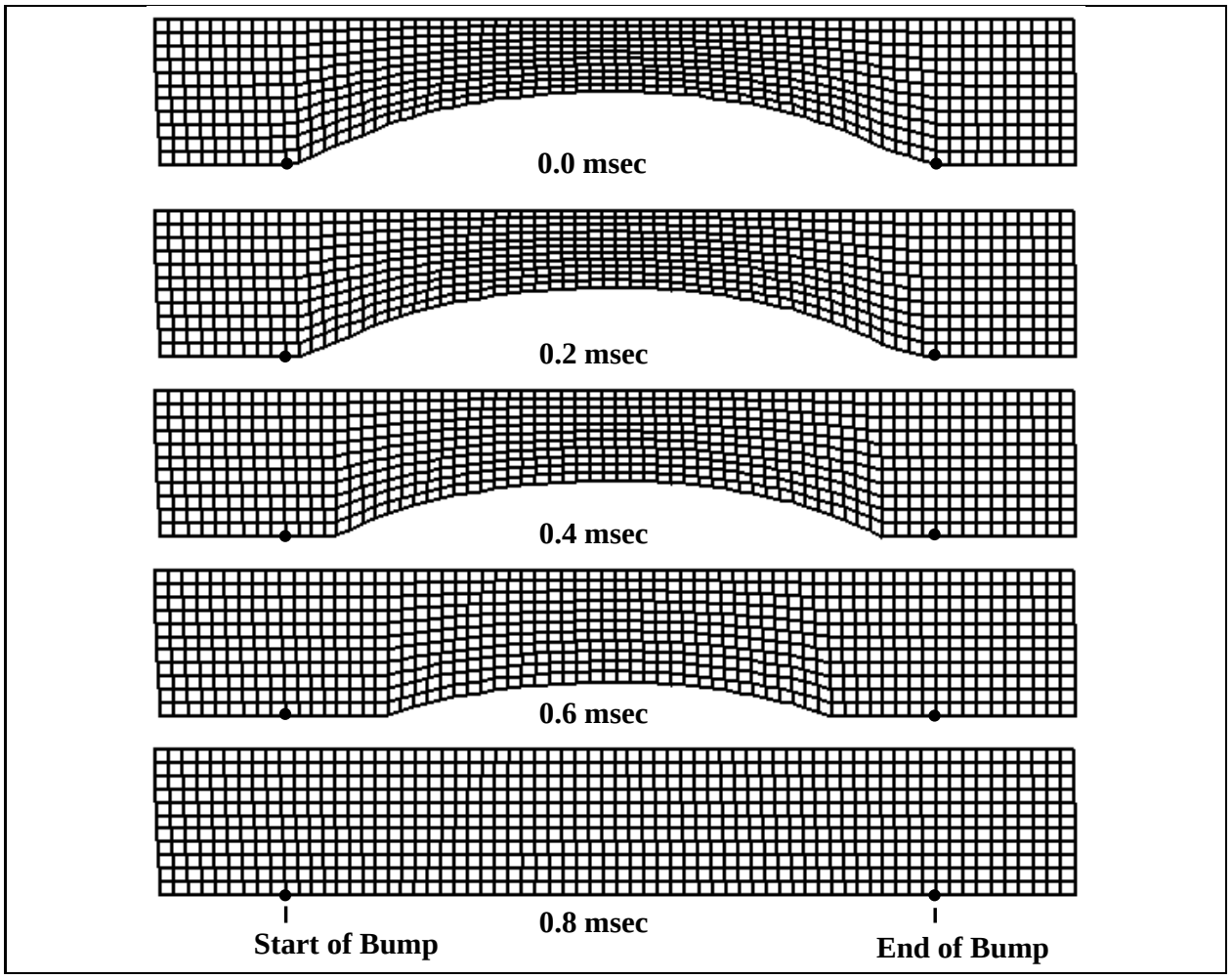

Figure 5: **The time history of the grid through the bump collapse (top-bottom: t=0.0 msec, t=0.2 msec, t=0.4 msec, t=0.6 msec, t=0.8 msec).**

References

[1] Anderson, D.A., J.C. Tannehill, and R.H. Pletcher. Computational Fluid Mechanics and Heat Transfer. New York: McGraw-Hill Book Company, 1984.

[2] Anderson, J.D., *Modern Compressible Flow*, McGraw Hill Inc., New York, 1984.

[3] Atwood, C.A., "An Upwind Approach to Unsteady Flowfield Simulation." AIAA 8th Applied Aerodynamics Conference, August 20-22, Portland, Oregon, 1990.

[4] Chung, Joongkee and G.L. Cole, "Comparison of Compressor Face Boundary Conditions for Unsteady CFD Simulations of Supersonic Inlet," AIAA-95-2627, July, 1995 (Also NASA TM 107194).

[5] Cooper, G.K. and Sirbaugh, J.R., "The PARC Code: Theory and Usage," AEDC-TR-89-15, December 1989.

[6] Freund, D.D., Sajben, M., and Slater, J.W., "Compressor-Face Boundary Condition Experiment: Generation of Acoustic Pulses in Annular Ducts," AIAA Paper 96-2657, July 1996.

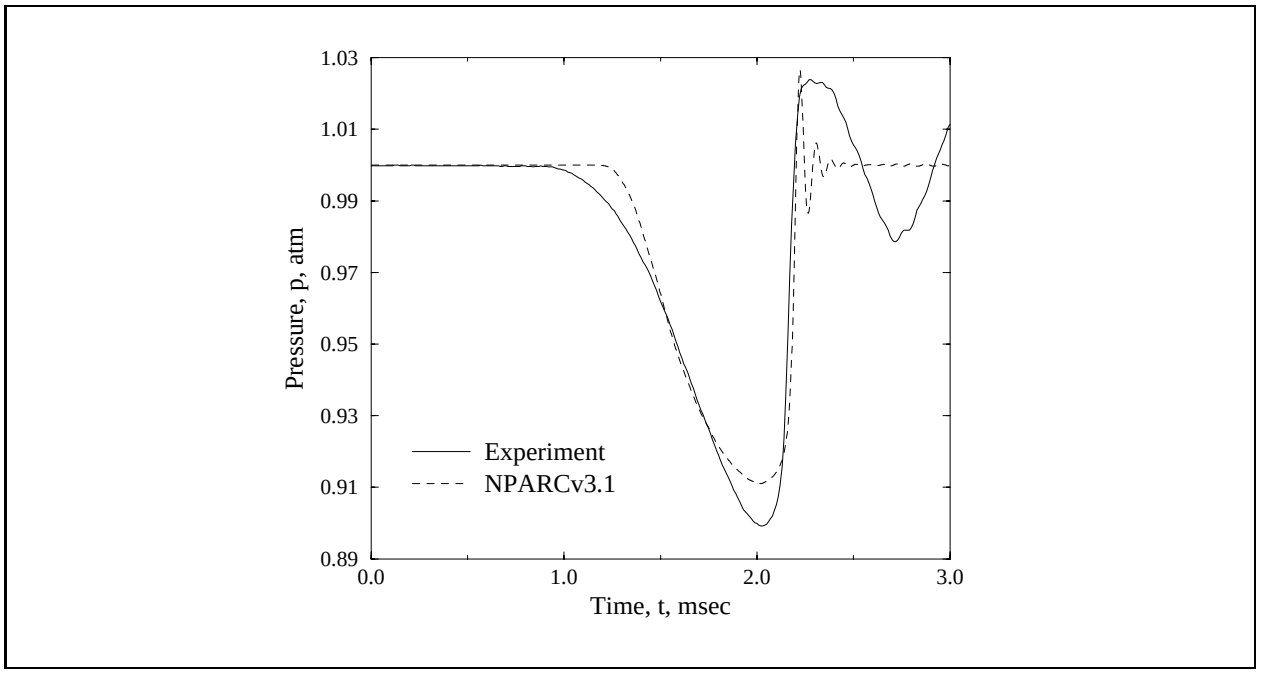

Figure 6: **The time history of the static pressure at sensor C for the 50% bump case.**

[7] Hirsch, C. Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows. New York: John Wiley & Sons, 1990.

[8] NPARC Version 3.0 User's Manual, September 1996.

[9] Mayer, D.W. and G.C. Paynter, "Boundary Conditions for Unsteady Supersonic Inlet Analyses," AIAA Journal, Vol. 32, No. 6, 1994, pp. 1200-1205.

[10] Paynter, G.C. and Mayer, D.M., "Evaluating the Temporal Accuracy of Inlet Normal Shock Propagation Simulations," *AIAA Journal*, **33**, 1534-36.

[11] Pulliam, T.H., "Time Accuracy and the Use of Implicit Methods," AIAA-93-3360, 1993.

[12] Rigby, D.L., "Compact Spatial Differencing and Subiteration Time Marching in the PARC Code," AIAA Paper 96-0385, January 1996.

[13] Slater, J.W., "Solving the Navier-Stokes Equations on Dynamic Grids," Unpublished Notes. May, 1997. (Download Postscript file at http://scratchy.lerc.nasa.gov/ fsjws/publications.html)

[14] Slater, J.W., "Study of CFD Methods Applied to Rapidly Deforming Boundaries," AIAA Paper 97-2041, June 1997.

[15] Sod, G.A., "A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws,"*Journal of Computational Physics*, **27**, 1-31.

[16] Thomas, P.D. and C.K. Lombard. "Geometric Conservation Law and Its Application to Flow Computations on Moving Grids." AIAA Journal 17 (1979): 1030-1037.

[17] Vinokur, M. "An Analysis of Finite-Difference and Finite-Volume Formulations of Conservation Laws." Journal of Computational Physics. 81 (1989): 1-52.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1999	Technical Memorandum

4. TITLE AND SUBTITLE

NPARC v3.1 User's Guide: A Companion to the NPARC v3.0 User's Guide

5. FUNDING NUMBERS

WU–509–10–51–00

6. AUTHOR(S)

Joongkee Chung, John W. Slater, Ambady Suresh, and Scott Townsend

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E–11594

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM—1999-209058
ICOMP–99–04

11. SUPPLEMENTARY NOTES

Joongkee Chung, Institute for Computational Mechanics in Propulsion, Cleveland, Ohio 44142 (work funded by NASA Cooperative Agreement NCC3–483); John W. Slater, NASA Glenn Research Center; Ambady Suresh and Scott Townsend, NYMA, Inc., 2001 Aerospace Parkway, Brook Park, Ohio 44142 (work funded by NASA Contract NAS3-98022, presently with Dynacs Engineering Corporation, Inc.). Responsible person, John W. Slater, organization code 5850, (216) 433–8513.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 34 and 61 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

NPARC v3.1 is a modification to the NPARC v3.0 computer program which expands the capabilities for time-accurate computations through the use of a Newton iterative implicit method, time-varying boundary conditions, and planar dynamic grids. This document discusses some of the changes from the NPARC v3.0, specifically: changes to the directory structure and execution, changes to the input format, background on new methods, new boundary conditions, dynamic grids, new options for output, usage concepts, and some test cases to serve as tutorials. This document is intended to be used in conjunction with the NPARC v3.0 user's guide.

14. SUBJECT TERMS

Computational fluid dynamics; Navier–Stokes equations

15. NUMBER OF PAGES

35

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	